International Journal of Mathematics and
Computer Applications Research (IJMCAR)
ISSN(P): 2249-6955; ISSN(E): 2249-8060
Vol. 5, Issue 2, Apr 2015, 25-38

FULL MULTIGRID METHOD WITH POLAR AND SPHERICAL POLAR TO CARTESIAN GRID TRANSFORM FOR SOLVING TWO AND THREE DIMENSIONAL ELLIPTIC PARTIAL DIFFERENTIAL EQUATION

OSAMA EL-GIAR

Department of Basic Science, Modern Academy for Engineering and Technology in Maadi, Cairo, Egypt

ABSTRACT

In this paper we use grids defined in Cartesian coordinates in place of grids defined in polar and spherical polar coordinates to solve an elliptic partial differential equation in two and three space dimensions. This transforming of the grids makes using both the interpolation and restriction operators more simple and moreover they give convergence rate better than operators defined in polar and spherical polar coordinates. A full multigrid method with $W(\gamma_1, \gamma_1)$ cycle with line relaxation and both restriction and interpolation operators defined in Cartesian form are used. Finally numerical examples have been given.

KEYWORDS: Multigrid Method, Numerical Analysis, Elliptic PDE, Cartesian, Polar and Spherical Polar Coordinate Systems

1 INTRODUCTION

Consider the two dimensional Poisson's equation defined in polar coordinate system:

$$U_{rr} + \frac{1}{r}U_r + \frac{1}{r^2}U_{\theta\theta} = -F(r\ ,\ \theta) \in \Omega$$

$subject\ to\ U = G\ on\ \partial\Omega$ where $\partial\Omega$ is a boundary of a quarter of a unit circle defined by:

$$\Omega = \{(r, 0):\ 0 \le r \le 1,\ 0 \le \theta \le \frac{\pi}{2}\} \tag{1.1}$$

and Consider the two dimensional general linear elliptic partial differential equation defined in polar coordinate system:

$$(1 + \sin\theta)\ U_{rr} + \frac{1}{r}(1 + \sin\theta\cos\theta)\ U_r + \frac{1}{r^2}(1 - \sin\theta\cos\theta)\ U_{\theta\theta}$$
$$+ \frac{1}{r^2}(\sin^2\theta - \cos^2\theta)\ U_\theta + \frac{1}{r}(\cos^2\theta - \sin^2\theta)\ U_{r\theta} = -F(r,\theta)\ \in \Omega$$

$subject\ to\ U = G\ on\ \partial\Omega$ where $\partial\Omega$ is a boundary of a quarter of a unit circle defined by

$$\Omega = \{(r, \theta):\ 0 \le r \le 1,\ 0 \le \theta \le \frac{\pi}{2}\} \tag{1.2}$$

also consider three dimensional Poisson's equation in spherical polar coordinate system:

$$U_{rr} + \frac{2}{r}U_r + \frac{1}{r_2}U_{\varphi\varphi} + \frac{1}{r_2}\frac{\cos\varphi}{\sin\varphi}U_{\varphi} + \frac{1}{r^2\sin^2\varphi}U_{\theta\theta} = -F(r,\varphi,\theta) \in \Omega$$

$$\text{subject to } U = G \text{ on } \partial\Omega$$

where $\partial\Omega$ is a boundary of a eighth of a unit ball defined by (1.3)

$$\Omega = \{(r,\ \varphi,\ \theta):\ 0 \le r \le 1,\ 0 \le \varphi,\ \theta \le \frac{\pi}{2}\}$$

2 MULTIGRID METHOD AND TRANSFORMATION OF THE GRIDS

For the multigrid method, we needs a sequence of grids, then replacing each term in equation 1.1 by the corresponding finite-difference approximation in the $r\ \theta$ -plane see [1] and [2]. In this paper we will solve these problems by using transformed grids defined in Cartesian coordinate system to overcome the difficulties in both interpolation and restriction operators. In the two dimensional space the difficulties of the interpolation operator is how to determine the interpolating values for points lying on the curved boundary and also for points lying in the neighborhood of the center of the circle of the fine grid and the difficulties of the restriction operator is how to calculate weights of the full-weight restriction operator see [1]. In the three dimensional space (in addition to the above difficulties of the interpolation operator) the difficulties of the interpolation operator is how to determine the interpolating values for points lying on the curved surfaces and also for points lying in the neighborhood of the lines passing through the center of the sphere and any of the poles of the sphere. In the transformed grids defined in Cartesian coordinate system we will use the a full multigrid method with $W(\gamma_1,\gamma_1)$ cycle, full weight residual restriction operator is used to get the residuals at coarse-grid points from the residuals computed at points on the fine grids. Also interpolation operator is used to get the approximation solution at fine-grid points from the solution computed at the coarse-grid points and even-odd line Gauss-Seidel relaxation is used. Numerical examples have been given.

3 FINITE DIFFERENCE DISCRETIZATON IN POLAR FORM AND IN SPHERICAL POLAR FORM

We will use the following finite difference approximation in the two dimensional space in polar form:

$$U_{rr} = h^{-2}(u_{i+1,j} + u_{i-1,j} - 2u_{i,j})$$

$$U_r = (2h)^{-1}(u_{i+1,j} - u_{i-1,j})$$

$$U_{\theta\theta} = (\delta 0)^{-2}(u_{i,j+1} + u_{i,j-1} - 2u_{i,j}) \qquad (3.1)$$

$$U_\theta = (2\delta\theta)^{-1}(u_{i,j+1} - u_{i,j-1}) \text{ and}$$

$$U_{r\theta} = (4h\delta\theta)^{-1}(u_{i+1,j+1} + u_{i-1,j-1} - u_{i+1,j-1} - u_{i-1,j+1})$$

$$F(r_i,\theta_i) = f_{i,j}$$

$$r = ih,\ i = 1,\ 2,\ ...,\ N$$

and we will use the following finite difference approximation in the three dimensional space in spherical polar form:

$$U_{rr} = h^{-2}(u_{i+1,j,k} + u_{i-1,j,k} - 2u_{i,j,k})$$

$$U_{r} = (2h)^{-1}(u_{i+1,j,k} - u_{i-1,j,k})$$

$$U_{\varphi\varphi} = (\delta\varphi)^{-2}(u_{i,j+1,k} + u_{i,j-1,k} - 2u_{i,j,k}) \tag{3.2}$$

$$U_{\varphi} = (2\delta\varphi)^{-1}(u_{i,j+1,k} - u_{i,j-1,k})$$

$$U_{\theta\theta} = (\delta\theta)^{-2}(u_{i,j,k+1} + u_{i,j,k-1} - 2u_{i,j,k})$$

$$F(r_i, \varphi_j, \theta_k) = f_{i,j,k}$$

$$r = ih,\ i = 1,\ 2,\ ...,\ N$$

Substitute each term of equation 3.1 in equation 1.1 we get the following system of equations in a matrix form given by:

$$\begin{bmatrix} -A & B & 0 & \dots & 0 \\ B & -A & B & \dots & 0 \\ 0 & B & -A & B & 0 \\ \dots & \dots & \dots & \dots & 0 \\ 0 & \dots & B & -A & B \\ 0 & \dots & 0 & B & -A \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_{N-1} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ b_{N-1} \end{bmatrix} \tag{3.3}$$

where $1 \le i, j \le N-1$, $h = \frac{1}{N+1}$, $\partial\theta = \frac{1}{2(N+1)}$ are the mesh size and

$$A = 2\left[1 + \frac{1}{(i\delta\theta)^2}\right] \tag{3.4}$$

$$B = \frac{1}{(i\delta\theta)^2}$$

where $1 \le i, j \le N-1$, u_{ij} is an approximation to the exact solution $U(r_i,\ \theta_j)$, b_{ij} is the right hand side, $h = \frac{1}{N+1}$, $\partial\theta = \frac{1}{2(N+1)}$, are the mesh sizes, where *N* is the number of interior points in each direction. Substitute each term of equation 3.1 in equation 1.2 we get the following system of equations in a

matrix form given by:

$$\begin{bmatrix} -A & B_1 & 0 & \dots & 0 \\ B_2 & -A & B_1 & \dots & 0 \\ 0 & B_2 & -A & B_1 & 0 \\ \dots & \dots & \dots & \dots & 0 \\ 0 & \dots & B_2 & -A & B_1 \\ 0 & \dots & 0 & B_2 & -A \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_{N-1} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ b_{N-1} \end{bmatrix} \tag{3.5}$$

where $1 \le i, j \le N-1$, $h = \frac{1}{N+1}$, $\partial\theta = \frac{1}{2(N+1)}$ are the mesh size and

$$A = 2\left[1 + \frac{1}{2}\sin 2\theta + \frac{1}{(i\delta\theta)^2}(1 - \frac{1}{2}\sin 2\theta)\right]$$

$$B_1 = \frac{1}{(i\delta\theta)^2}(1 - \frac{1}{2}\sin 2\theta) - \frac{\cos 2\theta}{2i^2\delta\theta} \tag{3.6}$$

$$B_2 = \frac{1}{(i\delta\theta)^2}(1 - \frac{1}{2}\sin 2\theta) + \frac{\cos 2\theta}{2i^2\delta\theta}$$

where $1 \le i, j \le N-1$, u_{ij} is an approximation to the exact solution $U(r_i, \theta_j)$, b_{ij} is the right hand side, $h = \frac{1}{N+1}$, $\partial\theta = \frac{1}{2(N+1)}$, are the mesh sizes, where *N* is the number of interior points

Also substitute each term of equation 3.2 in equation 1.3 we get the following system of equations in a matrix form given by:

$$\begin{bmatrix} -A & B & 0 & \dots & 0 \\ B & -A & B & \dots & 0 \\ 0 & B & -A & B & 0 \\ \dots & \dots & \dots & \dots & 0 \\ 0 & \dots & B & -A & B \\ 0 & \dots & 0 & B & -A \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_{N-1} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ b_{N-1} \end{bmatrix} \tag{3.7}$$

where, $1 \le i, j, k \le N-1$, $U_{i,j,k}$ is an approximation to the exact solution $U(r_i, \varphi_j, \theta_k)$, $b_{i,j,k}$ is the right hand side, , $h = \frac{1}{N+1}$, $\partial\varphi = \partial\theta = \frac{\pi}{2(N+1)}$ are the mesh sizes, and

$$A = 2\left[1 + \frac{1}{(i\delta\varphi)^2} + \frac{1}{(i\delta\theta\sin\varphi)^2}\right] \tag{3.8}$$

$$B = \frac{1}{(i\,\delta\theta \sin\varphi)^2}$$

where N is the number of interior points in each direction.

4 TRANSFORMATION OF THE GRIDS FROM A POLAR AND A SPHERICAL POLAR COORDINATES TO A CARTESIAN COORDINATE SYSTEM

The two dimensional domain defined in polar coordinates given by

$$\Omega = \{(r,\theta):\ 0 \le r \le 1,\ \ 0 \le \theta \le \frac{\pi}{2}\}$$

may be transformed into a domain defined in a Cartesian coordinates given by:

$$\Omega = \{(x,y):\ \ 0 \le x,y \le 1\}$$

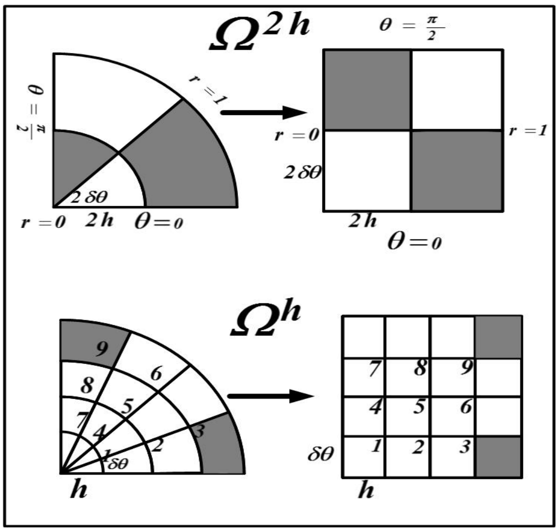

Figure 1: 2D-Transformation from a Polar to a Cartesian Grid

(see Figure 1)

also the three dimensional domain defined in spherical polar coordinates given by

$$\Omega = \{r,\varphi,\theta):\ 0 \le r \le 1,\ \ 0 \le \varphi,\ \ \theta \le \frac{\pi}{2}\}$$

may be transformed into a domain defined in a Cartesian coordinates given by:

$$\Omega = \{(x,y,z):\ \ 0 \le x,y,z \le 1\}$$

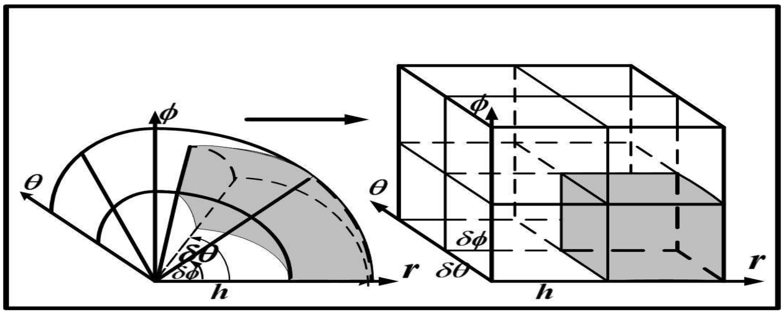

Figure 2: 3D-Transformation from a Polar to a Cartesian Grid

(see Figure 2)

5 FULL MULTIGRID METHOD

The elliptic equation have been solved using full multigrid method with $W\ (\gamma_1, \gamma_1)$ -cycle that has a sequence of 5 grids Ω_{kh} have been used with mesh sizes kh, k=2^n,n=0,1,…4, where $h = \frac{1}{64}$, with the coarse grid correction (CGC) and has the following components: see [2], [3] and see Figure 3

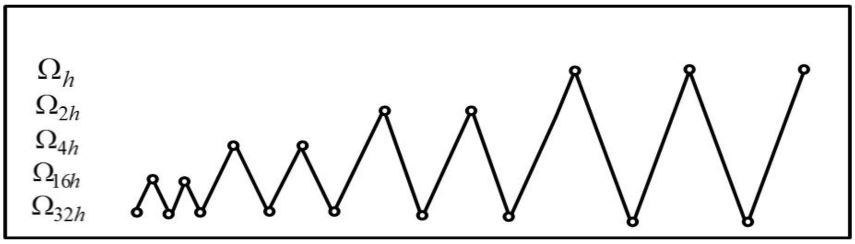

Figure 3: FMG $W\ (\gamma_1, \gamma_1)$ cycle

1. $W\ (\gamma_1, \gamma_1)$ Cycle with:

a) υ_1 is the number of relaxation sweeps before coarse grid correction (CGC).

b) Coarse grid correction (CGC).

c) υ_2 is the number of relaxation sweeps after coarse grid correction (CGC).

2. Red-Black Line Gauss-Seidel Relaxation: The most efficient smoothing iteration (relaxation) process is the red-black Gauss-Seidel iteration for lines which gives better results for polar problems than the point relaxation:

- For the two dimensional space the general grid can be defined by:

$$\Omega_k = \left\{ \left(\upsilon h_k ,\ \mu h_k \right),\ \upsilon,\ \mu \in Z\ ,\ Z\ \textit{is the set of integer numbers} \right\} \quad (5.1)$$

then split Ω_k into red (even) lines:

$$\Omega_k{}^a = \left\{ \left(\upsilon h_k ,\ \mu h_k \right),\ \in \Omega_k ,\ \ \upsilon\ \textit{is even} \right\} \quad (5.2)$$

and the remaining lines are black (odd):

$$\Omega_k^{\ b} = \left\{ \left(\upsilon h_k ,\ \mu h_k \right),\ \in \Omega_k ,\ \ \upsilon \text{ is odd} \right\} \tag{5.3}$$

- For the three dimensional space the general grid can be defined by:

$$\Omega_k = \{ (\nu h_k ,\ \mu h_k ,\ \gamma h_k),\ \nu,\ \mu,\ \gamma \in Z,\ \ Z \text{ is the set of integer numbers} \} \tag{5.4}$$

then split Ω_k into red (even) lines:

$$\Omega_k^a = \{ \nu h_k , \mu h_k , \gamma h_k) \in \Omega_k ,: (\nu + \mu) \text{ is even} \} \tag{5.5}$$

and the remaining lines are black (odd):

$$\Omega_k^b = \{ (\nu h_k , \mu h_k , \gamma h_k) \in \Omega_k ,: (\nu + \mu) \text{ is odd} \} \tag{5.6}$$

First the solutions are calculated at the points of the even (odd) lines using the solution at the points of the odd (even) lines and vise-versa, so for each line we have to use Gauss elimination method to solve a system of linear equation for equation 3.5 and for equation 3.7.

3. For Coarse to Fine Interpolation with Operator I_{2h}^{h} : It takes coarse-grid vector (solution or correction) u^{2h} defined on the coarse grid Ω^{2h} and produces fine grid vector u^h defined on the fine grid Ω^h according to the rule $I_{2h}^{h} u^{2h} = u^h$. the two and three dimensional space, we have the following components of u^h :

* For the two dimensional space the components of u^h are as follows:

$$u_{2i,2j}^{h} = u_{i,j}^{2h} \text{ for common points in the two grids (point 5)} \tag{5.7a}$$

$$u_{2i+1,2j}^{h} = \frac{1}{2}(u_{i,j}^{2h} + u_{i+1,j}^{2h}) \text{ for points in the } r = x \text{ direction, (points 4, 6)} \tag{5.7b}$$

$$u_{2i,2j+1}^{h} = \frac{1}{2}(u_{i,j}^{2h} + u_{i,j+1}^{2h}) \text{ for points in the } \theta = y \text{ direction, (points 2, 8)} \tag{5.7c}$$

$$u_{2i+1,2j+1}^{h} = \frac{1}{4}(u_{i,j}^{2h} + u_{i+1,j}^{2h} + u_{i,j+1}^{2h} + u_{i+1,j+1}^{2h}) \tag{5.7d}$$

for intermediate points, (points 1, 3, 5, 7)

see Figure 1

* For the three dimensional space the components of u^h are as follows

$$u_{2i,2j,2k}^{h} = u_{i,j,k}^{2h} \text{ for common points in the two grids} \tag{5.8a}$$

$$u^h_{2i+1,2j,2k} = \frac{1}{2}(u^{2h}_{i,j,k} + u^{2h}_{i+1,j,k})$$ for points in the $r = x$ direction (5.8b)

$$u^h_{2i,2j+1,2k} = \frac{1}{2}(u^{2h}_{i,j,k} + u^{2h}_{i,j+1,k})$$ for points in the $\varphi = y$ direction (5.8c)

$$u^h_{2i,2j,2k+1} = \frac{1}{2}(u^{2h}_{i,j,k} + u^{2h}_{i,j,k+1})$$ for points in the $\theta = z$ direction (5.8d)

$$u^h_{2i+1,2j+1,2k} = \frac{1}{4}(u^{2h}_{i,j,k} + u^{2h}_{i+1,j,k} + u^{2h}_{i,j+1,k} + u^{2h}_{i+1,j+1,k}) \tag{5.8e}$$

for points in the $r\,\varphi = xy$ plane direction

$$u^h_{2i+1,2j,2k+1} = \frac{1}{4}(u^{2h}_{i,j,k} + u^{2h}_{i+1,j,k} + u^{2h}_{i,j,k+1} + u^{2h}_{i+1,j,k+1}) \tag{5.8f}$$

for points in the $r\,\theta = xz$ plane direction

$$u^h_{2i,2j+1,2k+1} = \frac{1}{4}(u^{2h}_{i,j,k} + u^{2h}_{i,j+1,k} + u^{2h}_{i,j,k+1} + u^{2h}_{i,j+1,k+1}) \tag{5.8g}$$

for points in the $\varphi\theta = yz$ plane direction

$$u^h_{2i+1,2j+1,2k+1} = \frac{1}{8}(u^{2h}_{i,j,k} + u^{2h}_{i+1,j,k} + u^{2h}_{i,j+1,k} + u^{2h}_{i,j,k+1}$$
$$+ u^{2h}_{i+1,j+1,k} + u^{2h}_{i+1,j,k+1} + u^{2h}_{i,j+1,k+1} + u^{2h}_{i+1,j+1,k+1}) \tag{5.8h}$$

For intermediate points see Figure 2

4 Fine to Coarse Restriction Operator: The restriction operator denoted by I^{2h}_h takes the residual vector R^h computed on the fine-grid and transfer it to the coarse-grid according to the rule: $I^{2h}_h R^h = R^{2h}$, where the components of $R^{2h}_{i,j}$ of R^{2h} are given by two restriction operators: half-weight and full-weight restriction operator; since we use the line relaxation, then the full weight operator will be more effective than the half-weight operator see [2]and[3]:

- For the two dimensional space the full weight restriction operator R^{2h} will be defined by:

$$R^{2h}_{i,j} = \frac{1}{16}[R^h_{2i-1,2j-1} + R^h_{2i-1,2j+1} + R^h_{2i+1,2j-1} + R^h_{2i+1,2j+1} + \tag{5.9}$$
$$2(R^h_{2i,2j-1} + R^h_{2i,2j+1} + R^h_{2i-1,2j} + R^h_{2i+1,2j}) + 4R^h_{2i,2j}]$$

The matrix form of the full-weight operator I_h^{2h} takes the form:

$$I_h^{2h} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_h^{2h} \tag{5.10}$$

- For the three dimensional space the full weight restriction operator R^{2h} will be defined by:

$$R_{i,j,k}^{2h} = \frac{1}{2} R_{2i,2j,2k}^{h} + \frac{1}{4} S_1 + \frac{1}{8} S_2 + \frac{1}{16} S_3 \tag{5.11}$$

where

$$S_1 = R_{2i-1,2j,2k}^h + R_{2i+1,1j,2k}^h + R_{2i,2j-1,2k}^h + R_{2i,2j+1,2k}^h + R_{2i,2j,2k-1}^h + R_{2i,2j+1,2k+1}^h \tag{5.12}$$

$$S_2 = R_{2i-1,2j-1,2k}^h + R_{2i+1,2j-1,2k}^h + R_{2i-1,2j+1,2k}^h + R_{2i+1,2j+1,2k}^h + R_{2i,2j-1,2k-1}^h + R_{2i,2j+1,2k-1}^h + R_{2i,2j-1,2k+1}^h + R_{2i,2j+1,2k+1}^h + R_{2i-1,2j,2k-1}^h + R_{2i+1,2j,2k-1}^h + R_{2i-1,2j,2k+1}^h + R_{2i+1,2j,2k+1}^h$$

$$S_3 = R_{2i-1,2j-1,2k}^h + R_{2i+1,2j-1,2k-1}^h + R_{2i-1,2j+1,2k-1}^h + R_{2i-1,2j-1,2k+1}^h + R_{2i-1,2j+1,2k+1}^h + R_{2i+1,2j-1,2k+1}^h + R_{2i+1,2j+1,2k-1}^h + R_{2i+1,2j+1,2k+1}^h$$

The matrix form of the full-weight operator takes the form:

$$I_h^{2h} = \frac{1}{16}\left(\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} + \begin{bmatrix} 2 & 4 & 2 \\ 4 & 8 & 4 \\ 2 & 4 & 2 \end{bmatrix} + \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}\right)_h^{2h} \tag{5.13}$$

6 TEST PROBLEMS, RESULTS AND PERFORMANCE

In this section we report the results and performance of some test problems All our implementations in Fortran were executed on a Pentium 4 PC using FORTRAN90 workstation compiler. The accuracy of the method is measured by both L_2 norm of both defect and error and the max norm of both defect and the error.

6.1 Two Dimensional Poisson's Equation Defined in Polar Coordinate System

For the two dimensional Poisson's equation defined in polar coordinate system (equation 1.1), consider the following test problems:

1. Test Problem 1: $U_{exact} = e^{r^2 \sin\theta \cos\theta}$ (6.1)

2. Test Problem 2: $U_{exact} = \sin[r (3 \cos\theta + \sin\theta]$ (6.2)

3. Test Problem 3: $U_{exact} = \sin[r^2 \sin\theta + \sin\theta]$ (6.3)

Table 1: Fortran Implementation of Test Problem 1 Using *FMW* (2, 1) on PC

		Test Problem 1: Equation 6.1			
Test problem	**h**	**Max N(R)**	**L_2(R)**	**Max N(E)**	**L_2(E)**
Equation 6.1	$\frac{1}{2}$	0.232420D–07	0.232420D–07	0.389638D–01	0.389638D–01
	$\frac{1}{4}$	0.285286D–05	0.106780D–05	0.104190D–01	0.613996D–02
	$\frac{1}{8}$	0.111818D–04	0.395417D–05	0.265074D–02	0.132351D–02
	$\frac{1}{16}$	0.157221D–04	0.327736D–05	0.672460D–03	0.307192D–03
	$\frac{1}{32}$	0.922916D–04	0.492099D–05	0.167608D–03	0.487335D–04
	$\frac{1}{64}$	0.303821D–03	0.146161D–04	0.408888D–04	0.183232D–04

Table 2: Fortran Implementation of Test Problem 2 Using *FMW* (2, 1) on PC

		Test Problem 2: Equation 6.2			
Test problem	**h**	**Max N(R)**	**L_2(R)**	**Max N(E)**	**L_2(E)**
Equation 6.2	$\frac{1}{2}$	0.331118D–05	0.331118D–07	0.281695D–01	0.281695D–01
	$\frac{1}{4}$	0.488095D–05	0.234871D–05	0.132323D–01	0.814887D–02
	$\frac{1}{8}$	0.112966D–03	0.238629D–04	0.365061D–02	0.187780D–02
	$\frac{1}{16}$	0.185343D–03	0.195310D–04	0.932217D–03	0.439905D–03
	$\frac{1}{32}$	0.113296D–03	0.714172D–05	0.234723D–03	0.106309D–04
	$\frac{1}{64}$	0.501861D–04	0.203366D–05	0.580549D–04	0.256458D–04

Table 3: Fortran Implementation of Test Problem 3 Using *FMW* (2, 1) on PC

		Test Problem 3: Equation 6.3			
Test problem	**h**	**Max N(R)**	**L_2(R)**	**Max N(E)**	**L_2(E)**
Equation 6.3	$\frac{1}{2}$	0.479150D–08	0.479150D–08	0.139922D–01	0.139922D–01
	$\frac{1}{4}$	0.140393D–05	0.538168D–06	0.356133D–02	0.249522D–02
	$\frac{1}{8}$	0.875335D–05	0.349631D–05	0.892982D–03	0.533545D–03
	$\frac{1}{16}$	0.488703D–05	0.146127D–05	0.223801D–03	0.123341D–03
	$\frac{1}{32}$	0.302690D–05	0.661980D–06	0.557751D–04	0.297402D–04
	$\frac{1}{64}$	0.149890D–05	0.256758D–06	0.135154D–04	0.719024D–05

6.2 Two Dimensional General Elliptic Partial Differential Equation Defined in Polar Coordinate System

For the two dimensional general elliptic partial differential equation defined in polar coordinate system (equation 1.2), consider the following test problems:

1. Test Problem 4: $$U_{exact} = e^{0.5\, r^2 \sin\theta \cos\theta} + 2 r^2 \sin\theta \cos\theta \quad (6.4)$$

2. Test Problem 5: $$U_{exact} = e^{r^4 (\sin\theta \cos\theta)^2} \quad (6.5)$$

3. Test Problem 6: $$U_{exact} = r^6 (\sin\theta \cos\theta)^3 + 3 r^4 (\sin\theta \cos\theta)^2 + 6 r^2 (\sin\theta \cos\theta) \quad (6.6)$$

4. Test Problem 7: $$U_{exact} = \sin(r^2 \sin\theta \cos\theta + 1) \quad (6.7)$$

Table 4: Fortran Implementation of Test Problem 4 Using *FMW* (2, 1) on PC

		Test Problem 4: Equation 6.4			
Test problem	**h**	**Max N(R)**	**L_2(R)**	**Max N(E)**	**L_2(E)**
Equation 6.4	$\frac{1}{2}$	0.487803D–06	0.487803D–06	0.823281D–01	0.823281D–01
	$\frac{1}{4}$	0.293176D–04	0.179425D–04	0.130832D–01	0.681229D–02
	$\frac{1}{8}$	0.109685D–02	0.352481D–03	0.324392D–02	0.128157D–02
	$\frac{1}{16}$	0.117734D–02	0.234885D–03	0.942707D–03	0.273020D–03
	$\frac{1}{32}$	0.624465D–03	0.106234D–03	0.383377D–03	0.806606D–04
	$\frac{1}{64}$	0.561402D–03	0.452083D–04	0.144482D–03	0.258706D–04

Table 5: Fortran Implementation of Test Problem 5 Using *FMW* (2, 1) on PC

		Test Problem 5: Equation 6.5			
Test problem	**h**	**Max N(R)**	**L_2(R)**	**Max N(E)**	**L_2(E)**
Equation 6.5	$\frac{1}{2}$	0.340116D–06	0.340116D–06	0.355722D–01	0.355722D–01
	$\frac{1}{4}$	0.135265D–04	0.700796D–05	0.939620D–02	0.436022D–02
	$\frac{1}{8}$	0.738395D–04	0.267780D–04	0.237930D–02	0.977388D–03
	$\frac{1}{16}$	0.130248D–03	0.230902D–04	0.603557D–03	0.226562D–03
	$\frac{1}{32}$	0.780370D–04	0.114822D–04	0.159740D–03	0.543615D–04
	$\frac{1}{64}$	0.175214D–03	0.912809D–05	0.410080D–04	0.136098D–04

Table 6: Fortran Implementation of Test Problem 6 Using *FMW* (2, 1) on PC

		Test Problem 6: Equation 6.6			
Test problem	**h**	**Max N(R)**	**L_2(R)**	**Max N(E)**	**L_2(E)**
Equation 6.6	$\frac{1}{2}$	0.814372D–07	0.814372D–07	0.163969D+00	0.163969D+00
	$\frac{1}{4}$	0.418275D–04	0.190097D–04	0.299146D–01	0.153482D–01
	$\frac{1}{8}$	0.118158D–02	0.385504D–03	0.753403D–02	0.318098D–02
	$\frac{1}{16}$	0.138994D–02	0.263022D–03	0.199151D–02	0.694873D–03
	$\frac{1}{32}$	0.749245D–03	0.119769D–03	0.641346D–03	0.172192D–03
	$\frac{1}{64}$	0.478077D–03	0.449016D–04	0.208378D–03	0.462317D–04

Table 7: Fortran Implementation of Test Problem 7 Using *FMW* (2, 1) on PC

		Test Problem 7: Equation 6.7			
Test problem	**h**	**Max N(R)**	**L_2(R)**	**Max N(E)**	**L_2(E)**
Equation 6.7	$\frac{1}{2}$	0.150675D–06	0.150675D–06	0.673306D–02	0.673306D–02
	$\frac{1}{4}$	0.619946D–05	0.351387D–06	0.143331D–02	0.783644D–03
	$\frac{1}{8}$	0.457767D–04	0.147752D–04	0.354350D–03	0.187174D–03
	$\frac{1}{16}$	0.371336D–04	0.101259D–04	0.926256D–04	0.443316D–04
	$\frac{1}{32}$	0.492353D–04	0.546994D–05	0.267029D–04	0.110106D–04
	$\frac{1}{64}$	0.180558D–03	0.756322D–05	0.762940D–05	0.266461D–05

6.3 Three dimensional Poisson's Equation Defined in Spherical Polar Coordinate System: For the three dimensional Poisson's equation defined in spherical polar coordinate system: (equation 1.2), consider the following test problems:

1. Test Problem 8: $U_{exact} = r^6 \sin^4 \varphi \cos^2 \varphi \sin^2 \theta \cos^2 \theta + 1$ (6.8)

2. Test Problem 9: $U_{exact} = \sin(r^6 \sin^4 \varphi \ \cos^2 \varphi \ \sin^2 \theta \cos^2 \theta + 1)$ (6.9)

3. Test Problem 10: $U_{exact} = e^{r^6 \sin^4 \varphi \cos^2 \varphi \sin^2 \theta \cos^2 \theta + 1}$ (6.10)

Table 8: Fortran Implementation of Test Problem 8 Using *FMW* (2, 1) on PC

		Test Problem 8: Equation 6.8			
Test problem	**h**	**Max N(R)**	**L_2(R)**	**Max N(E)**	**L_2(E)**
Equation 6.8	$\frac{1}{2}$	0.813771D–06	0.813771D–06	0.400865D–02	0.400865D–02
	$\frac{1}{4}$	0.352586D–03	0.150108D–04	0.212562D–02	0.142138D–02
	$\frac{1}{8}$	0.143917D–03	0.932530D–04	0.674367D–03	0.456735D–03
	$\frac{1}{16}$	0.144340D–02	0.230529D–03	0.133991D–03	0.142577D–03

Table 9: Fortran Implementation of Test Problem 9 Using *FMW* (2, 1) on PC

		Test Problem 9: Equation 6.9			
Test problem	**h**	**Max N(R)**	**L_2(R)**	**Max N(E)**	**L_2(E)**
Equation 6.9	$\frac{1}{2}$	0.144218D–06	0.144218D–06	0.209641D–02	0.209641D–02
	$\frac{1}{4}$	0.185749D–03	0.790830D–04	0.108546D–02	0.728700D–03
	$\frac{1}{8}$	0.796397D–04	0.518679D–04	0.333011D–03	0.228682D–03
	$\frac{1}{16}$	0.144420D–02	0.205533D–03	0.701547D–04	0.725672D–04

Table 10: Fortran Implementation of Test Problem 10 Using *FMW* (2, 1) on PC

		Test Problem 10: Equation 6.10			
Test problem	**h**	**Max N(R)**	**L_2(R)**	**Max N(E)**	**L_2(E)**
Equation 6.10	$\frac{1}{2}$	0.120721D–05	0.120721D–05	0.111215D–01	0.111215D–01
	$\frac{1}{4}$	0.975143D–03	0.415379D–03	0.598264D–02	0.399128D–02
	$\frac{1}{8}$	0.396901D–03	0.257905D–03	0.193477D–02	0.130048D–02
	$\frac{1}{16}$	0.489924D–02	0.709457D–03	0.377893D–03	0.402946D–03

7 CONCLUSIONS

In the previous paper [1] there were some difficulties in determining both interpolation and restriction operators, in the two dimensional space the difficulties of the interpolation operator is how to determine the interpolating values for points lying on the curved boundary points and also for points lying in the neighborhood of the center of the circle of the fine grid and the difficulties of the restriction operator is how to calculate weights of the full-weight restriction operator see [1]. In the three dimensional space (in addition to the above difficulties of the interpolation operator) the difficulties of the interpolation operator is how to determine the interpolating values for points lying on the curved surfaces and also for points lying in the neighborhood of the lines passing through the center of the sphere and any of the poles of the sphere. In this paper we have used transformed grids defined in Cartesian coordinate system to avoid these difficulties, in addition we have found that the convergence rate is better than that with grids defined in polar and spherical polar coordinate system, this is because the round-off error depends on both $(i, \delta\theta)$ and $(i,, \delta\varphi, \delta\theta)$ while in Cartesian coordinate system the round-off error is independent of these factors.

REFERENCES

1. Osama El-Giar, "Two Full Multigrid Algorithms in Cartesian and Polar Coordinate Systems to Solve an Elliptic Partial Differential Equation with a Mixed Derivative Term in a Quarter of a unit Circle," *International Journal of Mathematics Computer Application Research (IJMCAR)*Vol. 4, Issue 6 pp. 75-86 December 2014.

2. Briggs L, "A Multigrid Tutorial," *Society for industrial and applied mathematics*,-2000.

3. Hackbucsh W, "Multigrid Methods and applications," *Springer Berlin Heidelberg Berlin*,-2003. *Society for industrial and applied mathematics*-2000.

4. W. L. Briggs, V. E. Henson, and S. F. McCormick, "A Multigrid Tutorial, Second Edition" SIAM Books, *Philadelphia* – 2000.

5. J. C. Tannehill, D. A. Anderson, *"Computational Fluid Mechanics and heat Transfer, Second Edition", Series in Computational and Physical Processes in Mechanics and Thermal Sciences*, - April 1, 1997.

6. J. E. Dennis and R. B. Schanbel, "Numerical Methods for Unconstrained Optimization and Nonlinear Equations", *SIAM Books*, *Philadelphia*, - 1996.

7. John D. Anderson, Jr. "Computational Fluid Dynamics: The Basics with Applications", *McGraw-Hill Inc*, - 1995.

8. G. D. Smith, "Numerical Solution of Partial Differential Equations: Finite Difference Methods.", *Oxford Applied Mathematics & Computing Science Series,* - January 16, 1986.
9. McCormick S. F, "Mulitigrid tutorial," *SIAM, Phildelphia-Pennsylvania* – 1987.
10. Briggs W. L, "Mulitigrid tutorial," *SIAM, Phildelphia-Pennsylvania*-1987.
11. Brandt A, "Multi-level adaptive technique (MLAT) for fast numerical solution to boundary value problem," *Springer, Berlin*-1